Département EEA - Faculté Sciences et Ingénierie

Master EEA SME

Projet BE

Année 2022-2023

La Ruche

Rédigé par : JIDAN Bilal, ERROUSSI Souleimane

16 mai 2023

Table des matières

1 Introduction

Dans le cadre de notre première année de master SME à l'université de paul sabatier, il nous est proposé un projet séparé en 2 partie à réaliser en 32h, un premier projet de base nous permettant de comprendre, nous familiariser avec un nouveau microcontrôleur(nucleo-476-rg) , nouveau logiciel(stm32cubeIDE) . et par la suite de réaliser un système qui correspond au projet de BE.

Notre projet BE est la Ruche connectée, permettant de connaître grâce au capteur de distance et et le capteur de température de connaître la température de la Ruche et permettant de surveiller la ruche si une personne s'approche de la ruche en envoyant ces informations dans un centre de surveillance grâce au capteur LoRa qui indique.

Pour réaliser ce bureau d'étude nous avons d'abord pris connaissance des capteurs (HC-SR04, DHT22) et de leur utilisation, du logiciel STM32cubeIDE et du capteur Grove LoRa. Ce projet a été réaliser par ERROUSSI souleimane et JIDAN Bilal.

2 NUCLEO-476-RG

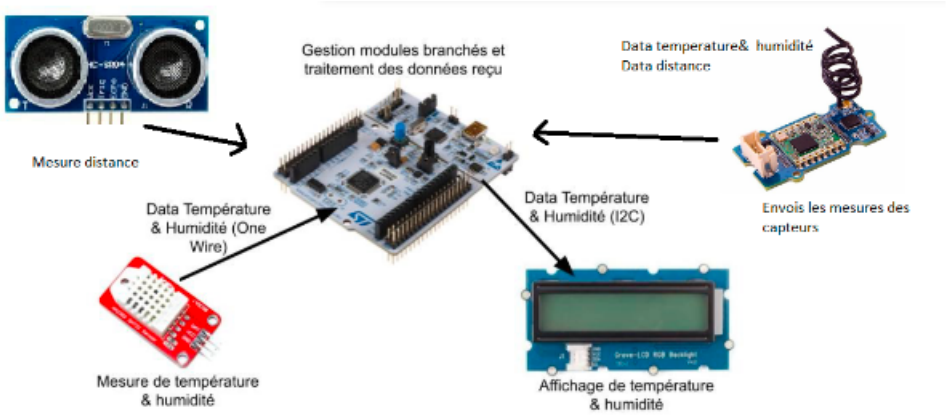
Pour commencer, nous devons configurer la communication avec l'écran LCD, qui utilise le protocole I2C à 4 fils. Sur le microcontrôleur, nous devons définir les broches SDA et SCL pour établir la communication avec l'écran LCD. Dans le schéma précédent, nous pouvons voir que les broches utilisées pour la communication sont PB8 et PB7.
En ce qui concerne le capteur de distance HC-SR04, il utilise un protocole de communication appelé One Wire. Pour permettre cette communication, nous devons attribuer une broche du microcontrôleur au capteur. Sur le schéma ci-dessus, nous pouvons observer que la broche utilisée est PA8. Selon la documentation technique (datasheet) du capteur, cette broche doit être maintenue à l'état logique 1 lorsqu'elle est au repos. Par conséquent, nous la configurerons en tant que pull-up.
Enfin, il est important de configurer un timer en microsecondes, car cela est nécessaire pour assurer le bon fonctionnement du capteur HC-SR04. Ainsi, en prenant en compte ces différentes configurations, nous pourrons établir une communication correcte entre le microcontrôleur STM32, l'écran LCD et le capteur HC-SR04.

Pour le capteur DHT22 on a utilise le microcontrôleur STM32L476rg communique avec le capteur DHT22 via le protocole I2C. Le microcontrôleur envoie une demande de mesure au capteur, qui répond ensuite avec les données de température et d'humidité. Une fois que le microcontrôcleur a récupéré les données, il les envoie à l'écran LCD pour les afficher. L'écran LCD est utilisé pour visualiser en temps réel les informations de température et d'humidité fournies par le capteur DHT22. Cette communication bidirectionnelle permet au microcontrôleur de contrôler le capteur DHT22 et de recevoir les données mesurées, puis de les afficher sur l'écran LCD pour une visualisation pratique et conviviale.

3 Schéma fonctionnel

Le microcontrôleur STM32L476rg communique avec le capteur DHT22 via le protocole I2C, le capteur HC-SR04 et le module LoRa selon le schéma suivant :

Le microcontrôleur envoie une demande de mesure au capteur HC-SR04, qui mesure la distance à un objet à l'aide d'ondes ultrasonores. Une fois que le capteur HC-SR04 a



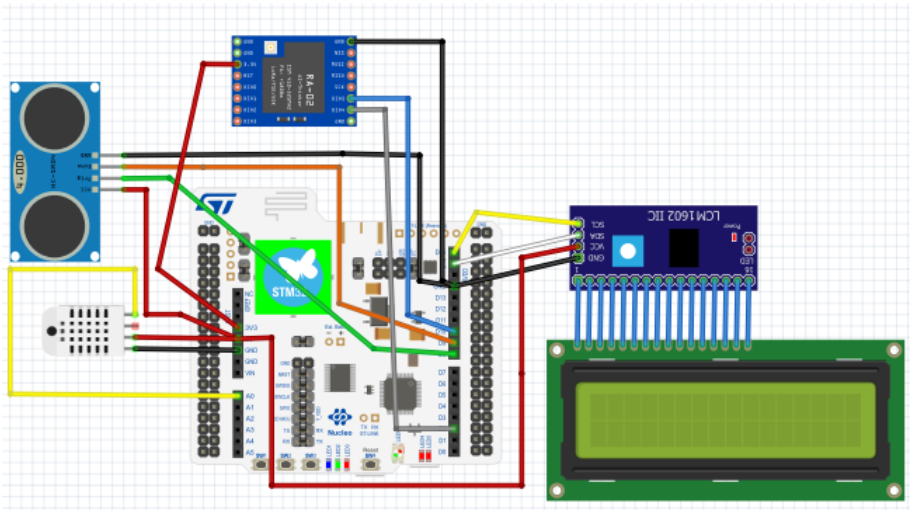
effectué la mesure, il envoie les données de distance au microcontrôleur STM32L476rg.

Parallèlement, le microcontrôleur envoie également une demande de mesure au capteur DHT22 via le protocole I2C. Le capteur DHT22 mesure la température et l'humidité et répond ensuite au microcontrôleur avec les données correspondantes.

En plus de cela, le microcontrôleur communique avec le module LoRa pour permettre une communication longue distance. Il peut envoyer les données collectées, telles que la distance, la température et l'humidité, via le module LoRa vers un autre dispositif distant, par exemple une passerelle LoRa ou un système de réception approprié.

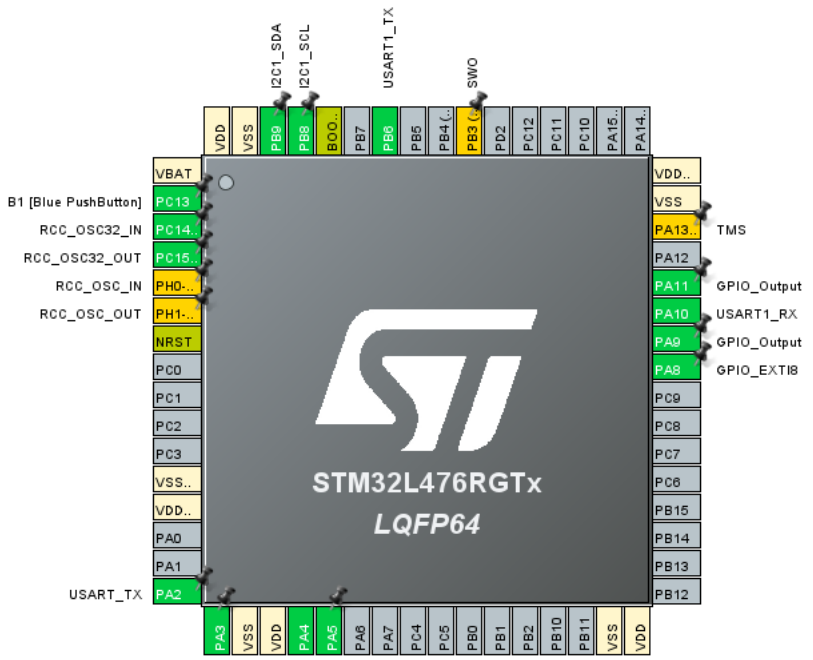
Une fois que le microcontrôleur a récupéré les données de distance du capteur HC-SR04, les données de température et d'humidité du capteur DHT22, il peut les envoyer via le module LoRa pour une transmission à distance. Les données peuvent être reçues par une passerelle LoRa ou un autre dispositif distant qui peut les traiter ou les afficher.

4 Schéma de câblage (fritzing)



5 Initialisation du microcontrôleur STM32L4

Avant toute chose, il faut d'abord paramétrer notre microcontrôleur selon notre utilisation.



Pour commencer, nous devons configurer la communication avec l'écran LCD, qui utilise le protocole I2C.

PA9 (GPIO output - broche Trig) : Cette broche est configurée en tant que sortie GPIO pour le triggeur du capteur ultrason. Elle envoie un signal d'ultrason lorsque le capteur doit mesurer une distance.

PA8 (GPIO input - broche Echo) : Cette broche est configurée en tant qu'entrée GPIO pour l'écho du capteur ultrason. Elle reçoit l'information renvoyée par le capteur après l'envoi de l'ultrason, permettant ainsi de mesurer la durée entre l'envoi et la réception de l'écho.

PA10 (USART1 input - broche RX) : Cette broche est configurée en tant qu'entrée USART1, spécifiquement pour la réception de données. Dans votre cas, elle est utilisée pour recevoir les données provenant du capteur ultrason (l'écho) via une communication série.

PB6 (USART1 output - broche TX) : Cette broche est configurée en tant que sortie USART1, spécifiquement pour la transmission de données. Dans votre cas, elle est utilisée pour envoyer les données du capteur LoRa via une communication série.

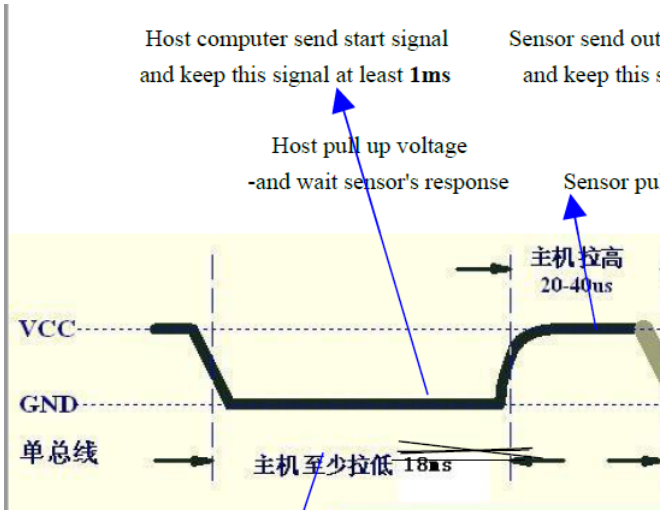
PA11 (GPIO output) : Cette broche est configurée en tant que sortie GPIO. On a mentionné qu'elle est utilisée pour le capteur de température dans la ruche. Cela signifie qu'elle envoie un signal pour lire les données de température et d'humidité à partir de ce capteur.

6 Initialisation du capteur DHT22

Nous allons commencer par créer une fonction pour notre temporisation en microsecondes, appelée "Delay_us".

```
void Delay_us(uint16_t us)
{
    __HAL_TIM_SET_COUNTER(&htim2, 0);
    while(__HAL_TIM_GET_COUNTER(&htim2) < us);
}
```

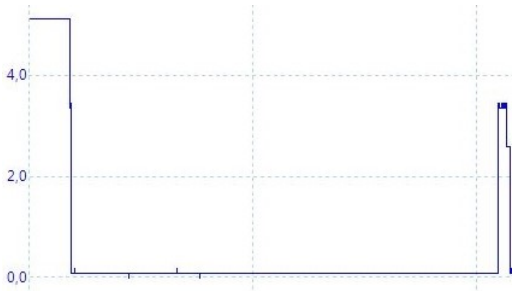
Ensuite, nous devons consulter la datasheet du capteur DHT22 pour comprendre comment il s'initialise.



D'après les informations de la datasheet du capteur DHT22, nous devons effectuer une séquence d'initialisation spécifique pour établir une communication réussie. Voici la procédure d'initialisation reformulée :

1 : Mettre la broche de signal reliant le microcontrôleur et le DHT22 à l'état logique bas (0) pendant au moins 1 milliseconde (ms) et au maximum 18 ms. Cela permet d'indiquer au capteur que le microcontrôleur est prêt à communiquer.

2 : Passé le délai d'attente initial, mettre la broche de signal à l'état logique haut (1) pendant une durée de 20 à 40 microsecondes (µs). Cette impulsion indique au capteur que le microcontrôleur est prêt à recevoir des données.



Lorsque nous comparons le graphe obtenu à l'aide du picoscope avec celui présent dans la datasheet, nous pouvons observer que notre séquence d'initialisation est en accord avec les spécifications.

Tout d'abord, nous constatons que notre broche de signal est à l'état logique 1 au repos, ce qui correspond à la spécification de la datasheet. Ensuite, nous effectuons une transition à l'état logique 0 et maintenons cette valeur pendant exactement 1,2 millisecondes (ms), ce qui correspond également à la durée spécifiée dans la datasheet.

Cependant, nous remarquons une légère variation de tension à droite du graphique, au moment où nous souhaitons revenir à l'état logique 0. Cette petite variation de hauteur est causée par le changement de direction du PIN, lorsque nous le passons en mode entrée. Cela peut entraîner une transition progressive de la tension avant qu'elle n'atteigne finalement l'état logique 0.

Il est important de noter que cette variation de tension n'affecte généralement pas de manière significative la communication avec le capteur, tant que nous respectons les délais et les niveaux logiques spécifiés dans la séquence d'initialisation.

```
Data_Output(GPIOA, GPIO_PIN_8); //info vers le capteur
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
DWT_Delay_us(1200); //signal de commande
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
DWT_Delay_us(30); //signal de commande
Data_Input(GPIOA, GPIO_PIN_8); //info vers le microcontroleur

/*commence la reception de donnees*/

while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8)));

for (k=0;k<1000;k++)
{
    if (HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_8) == GPIO_PIN_RESET)
    {
      break;
    }
}

while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8)));
DWT_Delay_us(40);
```

1 : Data_Output(GPIOA, GPIO_PIN_8) : Cette fonction configure la broche GPIOA_PIN_8 en tant que sortie, permettant ainsi d'envoyer des informations au capteur.

2 : HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET) : Cette instruction met la broche GPIOA_PIN_8 à l'état logique bas (0). Cela indique au capteur que le microcontrôleur est prêt à établir une communication.

3 : DWT_Delay_us(1200) : Cette fonction introduit un délai de 1200 microsecondes (1,2 millisecondes). Ce délai correspond à la spécification de la datasheet, où la broche doit être maintenue à l'état logique bas pendant 1,2 ms avant de passer à l'état logique haut.

4 : HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET) : Cette instruction met la broche GPIOA_PIN_8 à l'état logique haut (1). Cela indique au capteur que le microcontrôleur est prêt à recevoir des données.

5 : DWT_Delay_us(30) ; : Cette fonction introduit un délai de 30 microsecondes. Ce délai correspond à la spécification de la datasheet, où la broche doit être maintenue à l'état logique haut pendant 20 à 40 microsecondes.

6 : Data_Input(GPIOA, GPIO_PIN_8) : Cette fonction configure la broche GPIOA_PIN_8 en tant qu'entrée, permettant ainsi au microcontrôleur de recevoir les informations provenant du capteur.

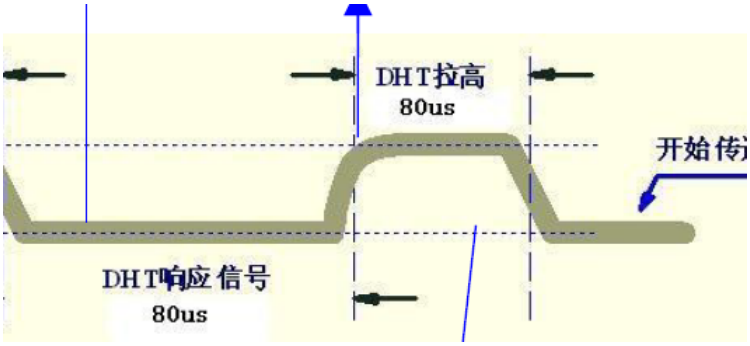
7 : while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8))) : Cette boucle attend que la broche GPIOA_PIN_8 passe à l'état logique haut, indiquant que le capteur est prêt à transmettre des données.

8 : for (k=0 ;k<1000 ;k++) : Cette boucle vérifie si la broche GPIOA_PIN_8 reste à l'état logique haut. Si elle passe à l'état logique bas, la boucle est interrompue.

9 : while(!(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8))) : Cette boucle attend que la broche GPIOA_PIN_8 revienne à l'état logique haut après avoir été à l'état logique bas. Cela indique que le capteur a terminé la transmission de données.

10 : DWT_Delay_us(40) : Cette fonction introduit un délai de 40 microsecondes. Ce délai correspond à une période d'attente après la fin de la transmission de données du capteur.

7 Réponse du DHT22



Si l'initialisation du capteur DHT22 est réussie, il répondra au microcontrôleur en envoyant un signal spécifique. Ce signal commence par une impulsion à l'état logique bas (0) qui dure environ 80 microsecondes (µs), suivie d'une impulsion à l'état logique haut (1) également d'une durée d'environ 80 µs. Ensuite, il y a une autre impulsion à l'état logique bas qui dure environ 50 µs. Après cette séquence, le capteur envoie un bit de donnée qui est à l'état logique haut (1) et marque le début de la transmission des données proprement dite.

8 Des calculs pour obtenir les valeurs d'humidité et de température

1 : Read_data(&dataH1) ; : Cette ligne de code appelle une fonction Read_data() pour lire le premier octet de données de l'humidité (dataH1) envoyé par le capteur DHT22. L'opérateur & est utilisé pour passer l'adresse de la variable dataH1 afin que la fonction puisse y stocker la valeur lue.

2 : Les lignes suivantes (Read_data(&dataH2) ;, Read_data(&dataT1), Read_data(dataT2) ;, Read_data(&SUM) ;) effectuent des opérations similaires pour lire les autres octets de données (dataH2, dataT1, dataT2) et la somme de contrôle (SUM) envoyés par le capteur

```
Read_data(&dataH1);
Read_data(&dataH2);
Read_data(&dataT1);
Read_data(&dataT2);
Read_data(&SUM);

check = dataH1 + dataH2 + dataT1 + dataT2;

RH = (dataH1<<8) | dataH2;
TEMP = (dataT1<<8) | dataT2;

Humidite = RH / 10.0;
Temperature = TEMP / 10.0;
```

DHT22.

3 : check = dataH1 + dataH2 + dataT1 + dataT2 : Cette ligne de code calcule la somme des octets de données (dataH1, dataH2, dataT1, dataT2) pour obtenir une valeur de vérification (check). Cette somme peut être utilisée pour vérifier l'intégrité des données lues. Il peut être comparé à d'autres calculs ou à une valeur attendue pour détecter les erreurs de lecture.

4 : RH = (dataH1<8) | dataH2 ; : Cette ligne de code combine les deux octets de données de l'humidité (dataH1 et dataH2) pour obtenir une valeur entière de l'humidité relative (RH). L'opérateur de décalage vers la gauche (<) est utilisé pour déplacer les bits de dataH1 de 8 positions vers la gauche, puis l'opérateur de "ou" binaire (|) combine les bits décalés avec ceux de dataH2 pour obtenir la valeur entière de l'humidité relative.

5 : TEMP = (dataT1«8) | dataT2 : Cette ligne de code combine les deux octets de données de la température (dataT1 et dataT2) de manière similaire à l'étape précédente, pour obtenir une valeur entière de la température (TEMP).

6 : Les lignes suivantes (Humidite = RH / 10.0, Temperature = TEMP / 10.0)s convertissent les valeurs de l'humidité et de la température en utilisant une division par 10.0. Cela permet d'obtenir les valeurs réelles de l'humidité et de la température avec une précision décimale.

Et on ajoute des lignes de programme concernant le capteur ultrason :

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{


    float distance=0;
    char chaine [50];
    __HAL_TIM_SET_COUNTER(&htim2,0);
    while(HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_8)==GPIO_

        distance=__HAL_TIM_GET_COUNTER (&htim2)/58;


    }
         sprintf(chaine,"distance=%2.f cm",distance
         lcd_position(&hi2c1,0,1);
         lcd_print(&hi2c1,chaine);
```

9 Caractéristiques capteur ultrason (HC-SR04)

Le capteur ultrason HC-SR04 a les caractéristiques suivantes :
1 : Dimensions : 45 mm x 20 mm x 15 mm
2 : Plage de mesure : 2 cm à 400 cm
3 :Résolution de la mesure : 0.3 cm
4 :Angle de mesure efficace : 15°

Largeur d'impulsion sur l'entrée de déclenchement : 10 s (Trigger Input Pulse width)
Ces caractéristiques décrivent les spécifications du capteur HC-SR04. Il a une petite taille, ce qui le rend pratique pour de nombreuses applications. Sa plage de mesure s'étend de 2 cm à 400 cm, ce qui signifie qu'il peut mesurer des distances allant de très courtes à relativement longues. La résolution de mesure de 0.3 cm indique la précision de la mesure effectuée par le capteur.

L'angle de mesure efficace de 15° indique la zone de couverture du faisceau ultrasonore émis par le capteur. Cela signifie que le capteur est plus sensible aux objets situés dans cette plage angulaire.

La largeur d'impulsion sur l'entrée de déclenchement est de 10 s. Cela fait référence à la durée de l'impulsion de déclenchement envoyée au capteur pour activer la mesure. Cette spécification est utile pour synchroniser le déclenchement de la mesure avec d'autres composants du système.

10 Fonctionnement du capteur ultrason (HC-SR04)

Le capteur ultrason HC-SR04 fonctionne de la manière suivante :

1 : Pour déclencher une mesure, il est nécessaire d'envoyer une impulsion de tension "high" d'au moins 10 s sur l'entrée "Trig" du capteur. Cela informe le capteur de commencer la mesure.
2 : Une fois l'impulsion de déclenchement reçue, le capteur émet une série de 8 impulsions ultrasoniques à une fréquence de 40 kHz. Ces impulsions sont envoyées dans une direction spécifique.
3 : Après avoir émis les impulsions ultrasoniques, le capteur attend de recevoir le signal réfléchi provenant d'un objet dans la zone de mesure.
4 : Lorsque le signal réfléchi est détecté, le capteur envoie un signal de tension "high" sur la sortie "Echo". La durée de ce signal est proportionnelle à la distance entre le capteur et l'objet.
5 : Le microcontrôleur connecté au capteur peut mesurer la durée du signal "Echo" et calculer ainsi la distance en utilisant une formule appropriée. En connaissant la vitesse du son dans l'air, qui est généralement de 343 m/s à une température de 20 °C, la distance peut être calculée en utilisant la formule : Distance = (Durée du signal Echo x Vitesse du son) / 2.

11 Distance de l'objet

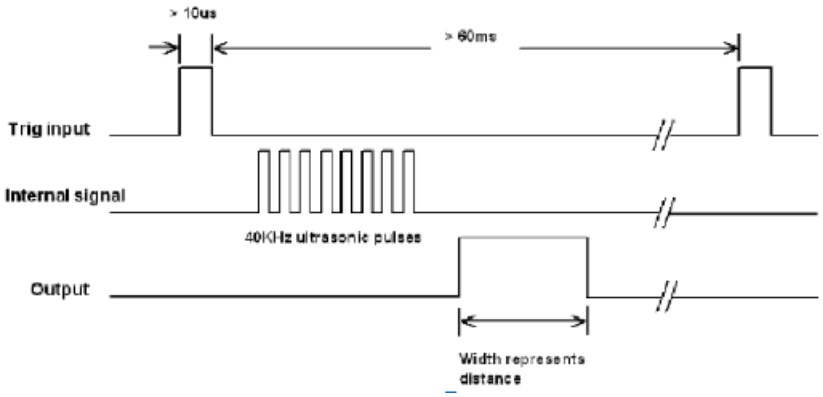
La distance parcourue par un son peut être calculée en multipliant la vitesse du son par le temps de propagation. La vitesse du son est d'environ 340 m/s (ou 34'000 cm/1'000'000

s). Le HCSR04, un capteur couramment utilisé, fournit une durée d'impulsion en dizaines de s. Pour obtenir le temps, il faut donc multiplier cette valeur par 10 s. Puisque le son effectue un aller-retour, la distance réelle correspond à la moitié de la distance calculée.

Donc, la formule simplifiée est : d = 17/100 cm * valeur.

Dans le manuel d'utilisation du HC-SR04, une autre formule est mentionnée : d = durée/58 cm. Cette formule est basée sur le fait que la fraction 17/1000 est équivalente à 1/58,8235. Cependant, elle produit des résultats moins précis.

Il convient de noter que, pour détecter un objet à une grande distance, sa surface doit mesurer au moins 0,5 m^2.



12 Programme du capteur Ultrason

La fonction HAL_GPIO_EXTI_Callback est une fonction de rappel (callback) qui est utilisée dans le cadre des interruptions des broches GPIO. Son rôle est d'être exécutée automatiquement par le microcontrôleur lorsqu'une interruption externe est déclenchée sur la broche GPIO spécifiée (GPIO_Pin).

13 Programme du capteur LoRa

Ce programme permet d'envoyer une information via le LoRa :

1 : Déclaration d'une variable de type pointeur de caractère appelée "cmdData" et attribution de la valeur "AT".

2 : Déclaration d'une variable de tableau de caractères appelée "cmdDataTmp" avec une taille de 40 caractères.

3 : Utilisation de la fonction sprintf pour formater la chaîne de caractères AT avec un retour à la ligne et stockage du résultat dans la variable "cmdDataTmp".

4 : Appel de la fonction HAL_UART_Transmit avec les paramètres suivants :
L'adresse de la structure "huart1".
Le tableau "charTransmit" contenant les données à transmettre (dans ce cas, probablement

```c
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    // Déclaration et initialisation des variables
    float distance = 0;
    char chaine[50];

    // Réinitialise le compteur htim2 à 0
    __HAL_TIM_SET_COUNTER(&htim2, 0);

    // Boucle tant que le GPIO pin PA8 est activé
    while(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8) == GPIO_PIN_SET)
    {
        // Calcule la distance en utilisant la valeur du compteur htim2 divisé
        distance = __HAL_TIM_GET_COUNTER(&htim2) / 58;
    }

    // Formate la valeur de la distance dans la chaîne chaine
    sprintf(chaine, "distance=%2.f cm", distance);

    // Positionne l'affichage LCD à la première ligne et à la première colonne
    lcd_position(&hi2c1, 0, 0);

    // Affiche la chaîne chaine sur l'affichage LCD
    lcd_print(&hi2c1, chaine);
```

"cmdDataTmp").
La valeur 1, indiquant le nombre de caractères à transmettre.
Un délai d'attente de 1000 millisecondes (1 seconde).

```c
uint8_t charTransmit[1];
char *cmdData= "AT";
char cmdDataTmp[40];
sprintf(cmdDataTmp,"%s\r\n",cmdData);

HAL_UART_Transmit(&huart1,charTransmit,1,1000);
```

14 Affichage des données obtenu sur l'écran LCD

```c
sprintf(bufRH,"Humidite: %.1f", Humidite);
sprintf(bufT, "Temp.: %.1f C", Temperature);
lcd_position(&hi2c1,0,0);
lcd_print(&hi2c1,bufRH);
lcd_print(&hi2c1,"%");
lcd_position(&hi2c1,0,1);
lcd_print(&hi2c1,bufT);
reglagecouleur(0,0,255);
```

Les lignes de code que vous avez fournies sont utilisées pour formater les valeurs d'humidité et de température et les afficher sur un écran LCD.

pour plus de detaille :

1 : sprintf(bufRH,"Humidite : %.1f", Humidite) : Cette ligne de code utilise la fonction sprintf pour formater la valeur d'humidité (Humidite) avec une précision décimale de 1 chiffre après la virgule. La valeur formatée est stockée dans la variable bufRH, qui est une chaîne de caractères. La chaîne de format "Humidite : %.1f" indique que la valeur de l'humidité sera insérée à la place de "%.1f" dans la chaîne.

2 : sprintf(bufT, "Temp. : %.1f C", Temperature) : Cette ligne de code fait la même chose que la précédente, mais cette foisci pour la valeur de température (Temperature). La chaîne de format "Temp. : %.1f C" indique que la valeur de la température sera insérée à la place de "%.1f" dans la chaîne, suivie de l'unité de mesure "C" pour Celsius.

3 : lcd_position(&hi2c1,0,0) : Cette ligne de code positionne le curseur de l'écran LCD à la première ligne (ligne 0) et à la première colonne (colonne 0).

4 : lcd_print(&hi2c1,bufRH) : Cette ligne de code affiche la valeur formatée de l'humidité (bufRH) sur l'écran LCD. La fonction lcd_print() est utilisée pour cela.

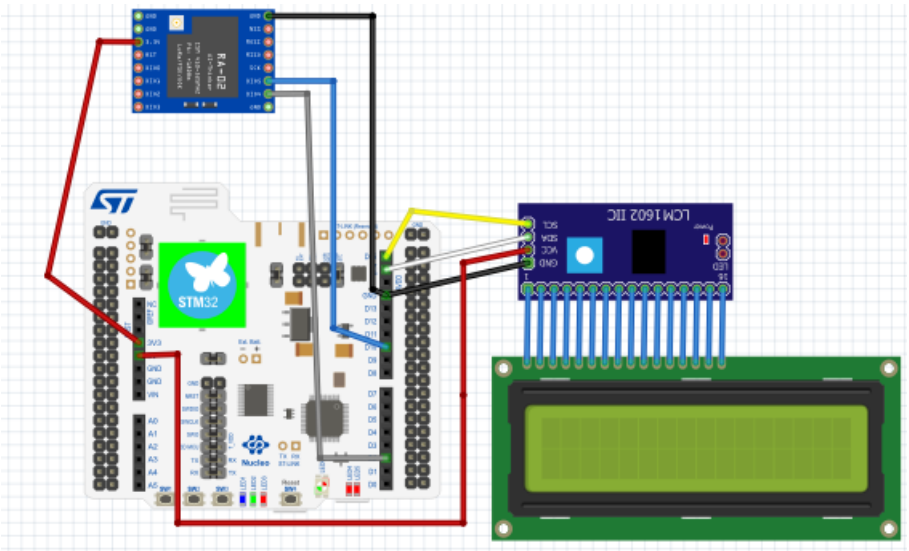
5 : lcd_print(&hi2c1,"%") ; : Cette ligne de code affiche le symbole "%" sur l'écran LCD. Il est utilisé pour indiquer l'unité de mesure de l'humidité.

6 : lcd_position(&hi2c1,0,1) ; : Cette ligne de code positionne le curseur de l'écran LCD à la deuxième ligne (ligne 1) et à la première colonne (colonne 0).

7 : lcd_print(&hi2c1,bufT) ; : Cette ligne de code affiche la valeur formatée de la température (bufT) sur l'écran LCD.

15 Reception des données avec un autre LORA à distance

voila un schema qui du montage :



le schema qui ci-dessus c'est pour recupere les donees et les informations a l'aide de l'autre Lora qui a ete brancher avec le STM32 avec les deux captures.

16 Resultat final du projet

Voici ci-dessous une photo du résultat final du projet, affichant les valeurs température et de la distance sur l'écran LCD :
voir la photo a la fin de rapport.

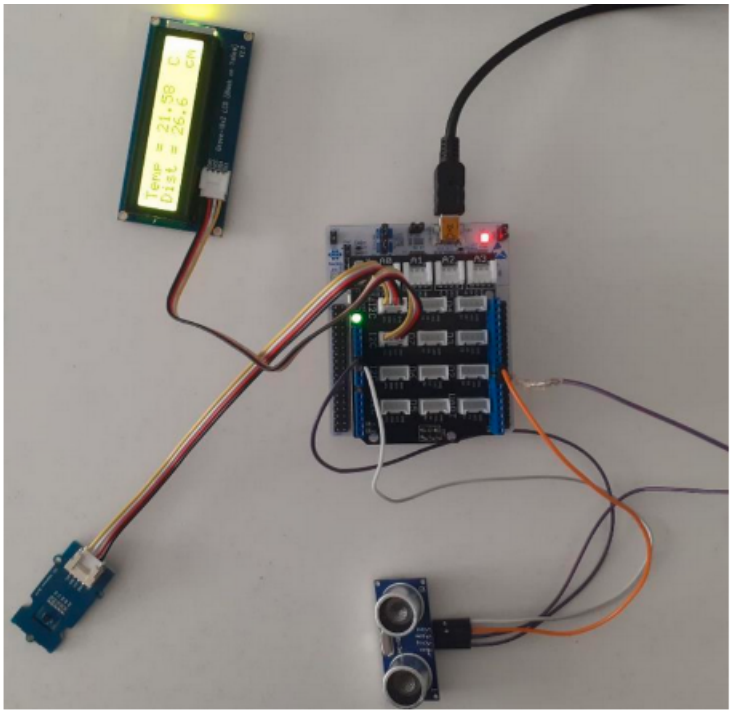
17 Conclusion

En conclusion, ce projet a été réalisé avec succès en utilisant le microcontrôleur STM32L476rg, le capteur de température et d'humidité DHT22, le capteur ultrason et le module LoRa. On a configuré les ports du microcontrôleur pour permettre la communication avec chaque composant, en utilisant des protocoles tels que l'I2C pour l'écran LCD, le One Wire pour le capteur DHT22 et les broches GPIO pour le capteur ultrason et le module LoRa.

En utilisant les spécifications et les fonctionnalités fournies par chaque composant, on a pu interagir avec eux pour collecter des données précises. Le capteur DHT22 a fourni des mesures fiables d'humidité et de température, tandis que le capteur ultrason a permis de mesurer avec précision la distance. Les données collectées ont ensuite été transmises via le module LoRa, permettant une communication à longue portée avec d'autre Lora.

Ce projet démontre l'interopérabilité et les capacités du microcontrôleur STM32L476rg, ainsi que la possibilité de combiner plusieurs capteurs pour obtenir des informations environnementales plus complètes. En ajoutant la technologie LoRa, il est possible d'étendre la portée de communication et d'interagir avec d'autres dispositifs distants.

Les applications de ce projet sont vastes, allant de la surveillance environnementale dans l'agriculture ou l'industrie à la mise en place de réseaux de capteurs pour la collecte de données à grande échelle. La flexibilité du microcontrôleur et la diversité des capteurs disponibles offrent de nombreuses possibilités pour des applications personnalisées selon les besoins spécifiques de chaque projet.


Temp = 21.50 C
Dist = 26.6 cm